REMARKS

Claims 1-29 are pending in the present application. Reconsideration of the claims is respectfully requested.

I. Response to Examiner's Rebuttal of Applicants' Arguments With Respect to Claims 1-29

The Examiner maintains his position as set forth in the Office Action dated October 27, 2003 with regard to the rejection of claims 1-29. Specifically, the Examiner maintains the rejection of claims 1, 2, 4-6, 11, 12, 14-16, 21, 22, 24-26 under 35 U.S.C. § 102(b) as being allegedly anticipated by *McManis* (U.S. Patent Number 5,692,047). Additionally, the Examiner maintains the rejection of claims 3, 10, 13, 20, and 23 under 35 U.S.C. § 103(a) as allegedly being unpatentable over *McManis* (U.S. Patent Number 5,692,047). Further, the Examiner maintains the rejection of claims 7-9, 17-19, and 27-29 under 35 U.S.C. § 103(a) as allegedly being unpatentable over *McManis* (U.S. Patent Number 5,692,047) in view of *Cordery et al.* (U.S. Patent Number 6,480,831). These rejections of claims 1-29 are respectfully traversed for at least the same reasons as noted in the Response to Office Action filed on January 27, 2004. Thus, since these rejections have been addressed in the Response to Office Action filed on January 27, 2004, Applicants will appropriately address only the Examiner's rebuttal of Applicants' arguments set forth in the Final Response to Office Action dated March 9, 2004 in the following remarks.

With regard to Applicants' arguments filed on January 27, 2004, the Final Office Action states:

> Applicant's arguments filed 1/27/04 have been fully considered but they are not persuasive. Applicant argues that McManis does not teach an automatically authenticated code with a first hash value embedded within in it. Examiner contends that McManis does teach a program executer receiving ANPrograms (authenticated code) with a hash value within (col. 7 line 25 thru col. 8 line 11) that a compiling party uses to verify the integrity of each ANProgram as it compiles it into an ASProgram (col. 11 line 61 thru col. 12 line 6).

Final Office Action dated March 9, 2004, page 2.

Independent claim 1, which is representative of the other rejected independent claims 11 and 21 with regard to similarly recited subject matter, reads as follows:

> 1. A method of verifying the integrity of unauthenticated code, comprising:
> receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code;
> receiving the unauthenticated code;
> generating a second hash value of the unauthenticated code;
> comparing the first hash value and the second hash value; and
> verifying the integrity of the unauthenticated code if the first hash value and the second hash value match. (emphasis added)

McManis does not teach or suggest the feature of receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code as recited in independent claims 1, 11 and 21 of the present invention. Additionally, *McManis* does not teach that the first hash value is hash value of unauthenticated code. The unauthenticated code is a separate and different entity than the automatically authenticated code.

McManis is directed towards a system and method in which a program interpreter for programs, whose integrity is verifiable, includes a facility for using non-verifiable programs from trusted sources. In addition, *McManis* provides a system and method for refusing to execute other non-verifiable programs. The *McManis* system includes a program executer that executes verifiable architecture neutral programs and a class loader that prohibits the loading and execution of non-verifiable programs unless the non-verifiable program resides in a trusted repository of such programs or the non-verifiable program is indirectly verifiable by way of a digital signature on the non-verifiable program that proves the program was produced by a trusted source.

Claim 1 recites receiving automatically authenticated code, wherein the automatically authenticated code includes an embedded first hash value of the unauthenticated code. A first hash value of unauthenticated code, such as for example native code, is embedded into automatically authenticated code, such as for example JAVA code that references the native code. The automatically authenticated code is received and the unauthenticated code is received; the automatically authenticated code and unauthenticated code are two separate entities. The method to verify the

cl. 8 lines 45-50

unauthenticated code comprises generating a second hash value from the received unauthenticated code and comparing the generated second hash value of the received unauthenticated code with the embedded first hash value from the automatically authenticated code. As stated previously, this first hash value that is embedded in automatically authenticated code is a hash value of unauthenticated code. *McManis* does not teach or suggest these features, as they are recited in claims 1, 11, and 21.

The Final Office Action states that *McManis* teaches a program executer receiving ANPrograms (authenticated code), with a hash value within, that a compiling party uses to verify the integrity of each ANProgram as it compiles it into an ASProgram. As shown in **Figure 2**, an ANProgram contains a digital signature of the originating party, which contains a message digest of the ANProgram code (column 5, lines 37-40). In other words, the ANProgram (authenticated code) contains a hash value of itself (authenticated code). To the contrary, the first hash value in claim 1 of the present application is a hash value of unauthenticated code. In claim 1, a separate unauthenticated code, such as native code called by an automatically authenticated code, is verified by comparing a first hash value of the unauthenticated code embedded in the automatically authenticated code to a second hash value generated from the received unauthenticated code. *McManis* does not teach or suggest the feature of receiving automatically authenticated code, the automatically authenticated code including an embedded first hash value of the unauthenticated code as recited in claims 1, 11 and 21.

The Final Office Action refers to the following portions of *McManis* in the rejection of claims 1, 11 and 21:

> The ANProgram compiler 162 then calls the signature generator 168 and instructs it to generate the ANProgram compiler's digital signature ($DigitalSignature_C$) 320 which can be verified to ensure that the ASProgram 300 was compiled with the trusted ANProgram compiler. This is done in a manner similar to that described earlier for generating the $DigitalSignature_{OP}$. However, in this case, the set of information signed is the ASProgram code and the $DigitalSignature_{OP}$. Another predetermined hash function with a corresponding $HashFunction_C$ ID 324 may be used to generate the message digest MD_C 322 of the set of information to be signed by the $DigitalSignature_C$, the private encryption key of the ANProgram compiler (Compiler's PrivateKey) is used to encrypt the MD_C and the $HashFunction_C$ ID, and the identifier of the ANProgram compiler (Compiler's ID) is added in clear text at the end of the encrypted MD_C

and $HashFunction_C$. The Compiler's PrivateKey and ID are provided by the ANProgram compiler.

The ANProgram compiler 162 calls the signature generator 168 a second time to generate the CompParty's digital signature ($DigitalSignature_{CP}$) 312, which can be verified by end users to ensure that the ASProgram 300 was generated by the trusted CompParty. This is done in a similar manner to that described earlier for generating the $DigtialSignature_{OP}$ (in the section discussing preparing an ANProgram for compiling). However, here the message digest (MD_{CP}) 314 generated for the $DigitalSignature_{CP}$ is generated by computing a predetermined or selected hash function ($HashFunction_{CP}$) on the data bits of the ASProgram code, the $DigitalSignatuer_{OP}$ and the $DigitalSignature_C$. Similar to the $HashFunction_{OP}$, for purposes of this disclosure, the $HashFunction_{CP}$ corresponds to the CompParty since it was used for the $DigitalSignature_{CP}$ of the CompParty.

The signature generator 168 then encrypts the MD_{CP} 314 and the ID of the $HashFunction_{CP}$ ($HashFunction_{CP}$ ID) 316 with the private encryption key of the CompParty (CompParty's PrivateKey). The signature generator then adds the identifier of the CompParty (CompParty's ID) 318 in clear text at the end of the encrypted items 314 and 316 to form the $DigitalSignature_{CP}$ 312. The CompParty's PrivateKey and ID are provided by the CompParty with the user interface 152.

After the $DigitalSignature_C$ 320 and the $DigitalSignature_{CP}$ 312 are generated, the ANProgram compiler 162 appends them to the ASProgram code 302, so that the resulting compiled ASProgram file or object has the following components in it:

ANProgram Code,
$DigitalSignature_{OP}$,
ASProgram Code,
$DigitalSignature_C$, and
$DigitalSignature_{CP}$.

McManis, column 7, line 25 through column 8, line 11.

> Similarly, the program executer has been described as requiring verification of both a $DigitalSignature_{CP}$ and a $DigitalSignature_C$. However, the program executer could be constructed to require verification of only one of these digital signatures and optionally verify the other digital signature if the ASProgram being verified includes it. Furthermore, the program executer could be constructed to skip the step of verifying the integrity of the ANProgram corresponding to each ASProgram, based on the assumption that the compiling party is trusted and that it is a duty of the compiling party to verify the integrity of each ANProgram that is compiles into an ASProgram prior to performing the compilation.

McManis, column 11, line 61 through column 12, line 6.

These portions of *McManis* only teach the steps of building a compiled ASProgram. The digital signatures for the compiler and compiling party are created and

appended to the ASProgram. The portions describe the contents of the ASProgram as shown in **Figure 3**. In the cited portions, *McManis* teaches authenticating digital signatures of the compiler and compiling party in an unauthenticated program (ASProgram). As discussed in the previous response, an ASProgram is unauthenticated code, i.e. the application specific program. *McManis* does not teach or suggest receiving automatically authenticated code, wherein the automatically authenticated code includes an embedded first hash value of the unauthenticated code as recited in independent claims 1, 11 and 21.

In these portions, *McManis* teaches that a compiled ASProgram is unauthenticated code, which contains ANProgram code, a digital signature of the originating party, ASProgram code, a digital signature of the compiler, and a digital signature of the compiling party. Thus, *McManis* teaches unauthenticated code may contain a digital signature of automatically authenticated code and a digital signature of unauthenticated code. However, claims 1, 11 and 21 recite that automatically authenticated code contains a hash value of unauthenticated code. Since a compiled ASProgram is unauthenticated code, *McManis* does not teach or suggest the features as recited in claims 1, 11 and 21 by describing the contents of a compiled ASProgram.

In view of the above, Applicants respectfully submit that *McManis* does not teach each and every feature of independent claims 1, 11, and 21 as required under 35 U.S.C § 102(b). Accordingly, Applicants respectfully request withdrawal of the rejection of claims 1, 11 and 21 under 35 U.S.C § 102(b). Further, Applicants respectfully request withdrawal of the rejection of dependent claims 2-10, 12-20 and 22-29 at least by virtue of their dependency on independent claims 1, 11, and 21, respectively.

II. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: May 10, 2004

Stephen J. Walder, Jr.
Reg. No. 41,534
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 367-2001
Attorney for Applicants

SJW/vja